A tooled methodology for the system architect's needs in simulation with autonomous driving application

Henri Sohier, Sahar Guermazi, Mouadh Yagoubi, Pascal Lamothe, Aldo Maddaloni, Pascal Menegazzi, Yining Huang

A Tooled Methodology for the System Architect's Needs in Simulation with Autonomous Driving Application

Henri Sohier*, Sahar Guermazi†, Mouadh Yagoubi*,
Pascal Lamothe‡, Aldo Maddaloni§, Pascal Menegazzi¶ and Yining Huang*

*_IRT SystemX_, Palaiseau, France - {firstname.lastname}@irt-systemx.fr
†_Sherpa Engineering_, La Garenne-Colombes, France - s.guermazi@sherpa-eng.com
‡_PSA Groupe_, Velizy-Villacoublay, France - pascal.lamothe@mpsa.com
§_Renault_, Guyancourt, France - aldo.maddaloni@renault.com
¶_Valeo_, Cergy-Pontoise, France - pascal.menegazzi@valeo.com

***Abstract*—Model Based Systems Engineering (MBSE) is a promising solution to formalize and communicate information about the design of complex systems, in particular for the automotive industry which faces new challenges associated to autonomous driving. Numerical simulation is commonly used to support the design of these complex systems, but the possible relations with MBSE should be further investigated. This work, conducted with academic and industrial partners at the research institute IRT SystemX, aims at further bridging the gap between system architecture and numerical simulation. An industrial design problem related to the design of an autonomous vehicle passing traffic lights is used to validate and illustrate new methods and tools based on SysML. Their aim is to: 1) guide the formulation of a question requiring simulation, called solicitation, by the system architect 2) guide the design of a simulation architecture by the simulation architect with a special focus on the consistency with the system. A Java plugin was developed in the SysML editor Papyrus for the solicitation, and a SysML metamodel was defined for the simulation architecture. The solicitation associated to the industrial design problem is answered by a multiobjective optimization of the vehicle's cost and electrical consumption using a co-simulation between the tools Simulink and Amesim.**

***Index Terms*—MBSE, SysML, System architect, Solicitation, Simulation architect, Model Identity Card**

I. Introduction

A. Context

The research institute IRT SystemX gathers industrial and academic partners to complete projects which generally last three to four years. One of the main objectives of the project AMC is to bridge the gap between systems engineering on one hand and complex simulations on the other hand by defining a step-by-step methodology supported by new software functions. Simulation is considered itself a potential complex system which can have a different topology from the original system it represents, especially because of the implementation of physics phenomena. It also includes different specifications from the system, typically regarding the accuracy or the speed of the simulation models. In this case, the role of simulation architect, which has recently emerged ([1]), seems to be necessary to specify the simulation. As shown in Fig. 1, the left branch of the idealized system V-model can lead to multiple needs in terms of simulation, at any step. A simulation architect can answer these needs by designing and integrating the simulations which will provide the results necessary for the system, following idealized simulation V-models. The industrial partners involved on this topic in the project AMC include large companies from the automotive and aeronautics industry (PSA Groupe, Renault, Valeo, Airbus) as well as software developers and service providers (Sherpa Engineering, Digital Product Simulation, Siemens). In the project AMC, a practical design problem allows the different partners first to identify the issues and lacks in the current design processes, and then to test possible solutions. This design problem concerns the early design of an autonomous vehicle which can pass traffic lights. It is presented in section II and then used to illustrate each proposition.

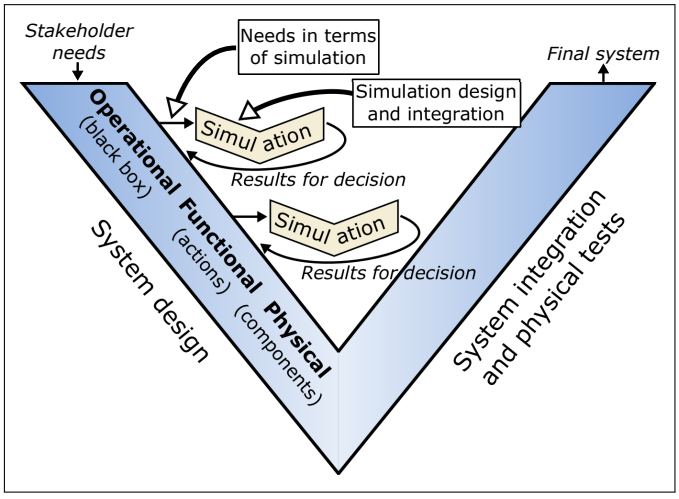

Fig. 1. Use of the simulation in a system V-model.

B. Today's role of systems engineering and simulation

Companies show a growing interest for systems engineering which aims at formalizing the multidisciplinary design of complex systems offering multiple services to the user. It is especially true in the automotive industry where new services gradually improve the autonomous driving capacity of the cars, from parking to lane keeping or platooning. Autonomous driving sheds light on the control systems which have been historically designed with systems engineering-like approaches ([2]). In the general landscape of systems engineering, Model-Based Systems Engineering (MBSE) appears as particularly promising. In a model, diagrams are easier to formalize than text, and they are an efficient way to communicate. MBSE is under maturation and efforts are now underway to offer user-friendly tools, a clear design methodology, and a modeling language at the right level of complexity. Regarding the tools, the SysML editor Papyrus has an open-source code which forms a great basis for new developments ([3]) in spite of a relatively complex interface and certain bugs. Some commercial tools like MagicDraw have customers in the automotive industry like the car manufacturer Renault ([4]). New tools or new plugins are also made available by startup companies whose funders often have experience in the automotive industry ([5], [6]). However, the MBSE tools are generally still not the design backbone they could be. Regarding the methodology, different approaches exist and some of them are gaining in importance, often by means of courses and certifications ([7], [8]), some of which are specifically addressed to the employees of car manufacturers like PSA Groupe ([8]). Some approaches aim at partially automatizing the design process through the reuse of functions and components ([9]). Finally, regarding the language, SysML can be considered as relatively hard to use, which can lead to a new organizational silo in the company. However, SysML is a well-known, rich standard which is often taken as a reference or as a source of inspiration, like for the language used in the tool Capella ([10]).

Although still relatively independent from systems engineering, simulation is now a common tool in most companies. The simulation development and the system development can be carried out by the same experts, but large companies can distinguish these tasks. PSA Groupe organized the simulation in groups adapted to different physics, from vibration to combustion and electricity, but also to computing issues such as optimization, uncertainties and model reduction. The resulting simulations are sometimes labeled as "functional", "multi-body", or "finite elements" ([11]). Renault started a division called "Model Factory" with hundred people in France, India and Korea dedicated to the simulation ([12]). The representativeness of the simulation, i.e. its likeness with the system it represents, is still often hard to quantify and can limit some decisions. In spite of its limits, the use of simulation is expected to grow with the development of autonomous vehicles. Indeed, the validation of their reliability may need billions miles of driving ([13]) which could be partially virtualized.

C. Gap between systems engineering and simulation

In systems engineering, some of the questions raised by the design process can be answered thanks to simulation. The link between systems engineering and simulation is sometimes very direct, like in the tool STIMULUS ([14]) where formal textual requirements and state diagrams can be tested thanks to random variables, or like in the tool CIL4Sys ([5]) where formal sequence and state diagrams typically representing control laws can be tested thanks to on-the-shelf or simple plant simulation models and user-defined environments. However, finer and more complex simulation models can be required for a better representation of the effects of control laws, or to test a physical solution. In this case, it becomes more difficult to mix the system and the simulation as they generally start to involve different people and have different topologies. Furthermore, while experts in control have the ability to make the necessary formal diagrams, MBSE can be considered as semi-formal or even informal ([15]), which limits such automations.

Thus, the design process can raise questions which can only be answered by simulation experts. There is currently little help to identify the information to communicate to these experts as well as the format to use. How to describe the system or the part of the system to simulate, the level of detail, the environment scenarios, or the simulation objectives? MBSE, which facilitates the communication of system specifications, should be used as a basis to formulate the questions about the system. New methodology, metadata and tool based on SysML are first presented in the section III dedicated to the solicitation from the system architect.

This solicitation represents the simulation needs. It is used to design the simulation means respecting the desired quality, cost, and delivery. The corresponding architecture tasks are considered to be carried out by a simulation architect. For example, the simulation architect has to find existing simulation models as well as pre- and post-processing functions in a cross-disciplinary approach given the expected accuracy or the costs of acquisition and execution. He also has to specify the missing simulation models and functions, and to check whether the simulation models correctly represent the system. New methodology and metadata guiding the architecture of the simulation are presented in section IV.

Thus, the simulation architecture is distinguished from the system architecture. This separation appears in [16] which refers to a "domain model" describing the system and to an "analysis model" describing the simulation, with block-to-block relations. However, these relations are mainly based on a generic simulation block and cannot alone show the complexity of the simulation architecture. In many works, simulation is seen as an enriched version of the system and the distinction between the system and the simulation is more tenuous. In [17], the simulation includes a system and simulation parameters. In [18] and [19], the simulation is based on blocks called "Analyzable Product Models" dedicated to design properties and on blocks called "Analysis Building Blocks" dedicated to generic equations like physical laws.

In several works, a SysML architecture is converted into a runable format, like in [20] which focuses on parametric diagrams, or in [21] and [22]. Depending on the approach, additional modeling can be required in the runable format.

The remainder of this paper is organized as follows. The industrial design problem of the project AMC is first presented in section II; New methodology, metadata and tool for the solicitation from the system architect for a new simulation are then presented in section III; New methodology, metadata and tool specification for the design of a new simulation architecture answering the needs of the system architect are presented in section IV; Finally, the simulation obtained for the industrial design problem and its results are presented in section V.

II. Industrial design problem

The industrial design problem used in the project and in this paper is related to the design of an autonomous vehicle passing traffic lights. A SysML model has first been developed with the tool PhiSystem based on Papyrus and provided by the partner Sherpa. It represents the complete car, with a special focus on the ability to pass traffic lights. The process followed is a synthesis of the different partners' approaches in systems engineering.

As represented in Fig. 2, the stakeholders are first identified, from marketing to logistics or regulations. The needs in terms of function, performance or constrain are then reviewed for each stakeholder. For example, the marketing requires to have an electrical consumption and a noise below certain thresholds. The needs are finally organized in services, like "autonomous driving", "acoustics" and "energy".

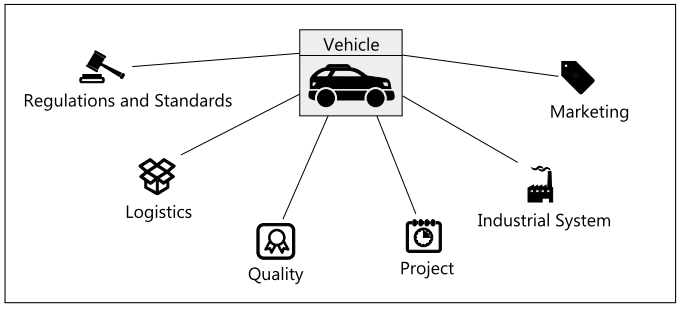

Fig. 2. Stakeholders of the considered industrial design problem.

The system design is then carried out at three different levels. First, in the operational level, the system is described for each service as a black box. The use cases permit to list for each service the systems expected capacities, like "staying on a lane", "passing the traffic lights" or "following a car". For each use case, the necessary exchanges between the system and the environment actors are defined in operational sequence diagrams. Operational activity diagrams help to specify the behavior of the system between these exchanges. Each operational sequence diagram is associated to a single operational state like "active", "non active" or "fault". All the operational states are summarized in an operational state diagram.

The operational sequence diagrams, where the system is a black box, are then transformed into functional sequence diagrams where the system is divided into functions like "Control the vehicle" or "Provide power". Likewise, each function can then be divided into sub-functions in more functional sequence diagrams. All the functions, sub-functions and functional exchanges interconnecting them are summarized in a multi-layer internal block diagram. The first layer is represented in Fig. 3. Functional activity diagrams can specify the behavior of the functions used in the functional sequence diagrams. Fig. 4 shows an activity diagram for the control function. If there is a traffic light within a certain range, the control function can output three different commands: 1) In "Speed envelop", the vehicle goes at the maximum safe speed 2) In "Pass", the vehicle keeps its current speed to pass the traffic light 3) In "Stop", the vehicle breaks to stop. The "time to light" is the ratio of the distance between the vehicle and the traffic light, to the current speed of the vehicle. Two properties can be tuned: 1) the range from which the traffic light starts to be taken into account 2) the shape of the speed envelop, defined by the maximum acceptable deceleration. These control properties affect the electrical consumption of the vehicle.

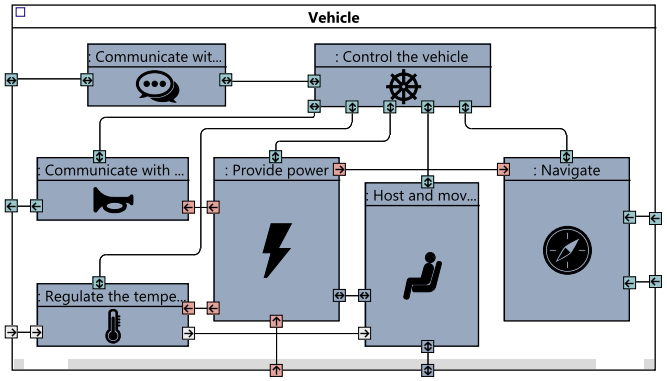

Fig. 3. Internal block diagram at the functional level.

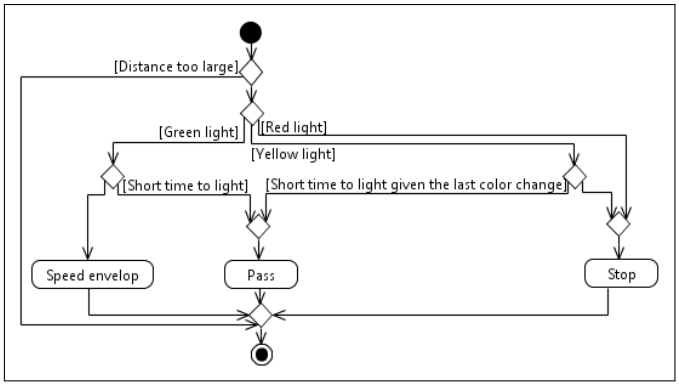

Fig. 4. Activity diagram of the control function for the traffic lights.

The last layer of sub-functions is finally allocated to components like "Controller" or "Powertrain" which are also represented in a multi-layer internal block diagram. The detection of

the traffic light is carried out by a component called “Sensor”. One of its properties represents the uncertain distance from which the traffic light is detected. The better this property is, the more expensive the sensor is. The properties of the functions and components can be associated to requirements. Fig. 5 represents a simplified version of the data model of the system architecture.

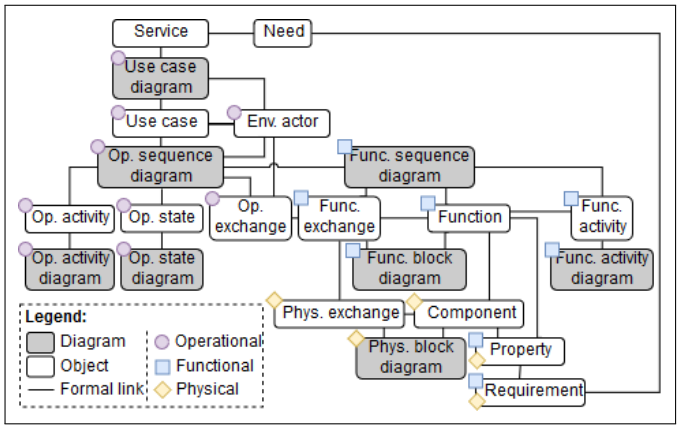

Fig. 5. Simplified data model of the system architecture.

In this context, it is considered that the system architect is looking for the best control properties and the best sensor properties to minimize both the cost and the electrical consumption when passing traffic lights.

III. Solicitation from the system architect

The solicitation from the system architect is a documented question. It describes the needs in terms of simulation, independently from any choice in terms of simulations means. Thus, the system architect does not handle simulation models. In the design problem considered, the solicitation is the documented question regarding the choice of control and sensor properties to minimize both the cost and the electrical consumption when passing traffic lights.

A. Filtered system architecture

If a question raises during the development of a system, the system architect can formalize it thanks to the model-based system architecture. Indeed, it precisely aims at facilitating the communication of the system specifications which should be used for the development of the simulation. However, the system architect should not communicate the whole system architecture every time a question raises. For example, the system architect should not communicate information about the color of the windows control buttons if he wants a fast answer regarding the electrical consumption of the vehicle when passing traffic lights. Similarly, the system architect should not communicate precise details about the sensor design if he wants a general simulation which does not take them into account. The system architect has the global understanding of the system which is required for such filtering, and he can limit the information before it is shared with the simulation developers.

The system architect needs a methodology and a tool to filter the information. The tool is prototyped as a Java plugin in PhiSystem and Papyrus. The following paragraphs focus on the tool and its graphical interface as they validate the underlying methodology and permit to understand it from a user point of view. As shown in Fig. 6, a menu “Solicitation for simulation” is added to the top bar. A sub-menu first permits to copy the system architecture. This copy is an opportunity to select a specific version and configuration of the system, and to later modify or simplify some specifications of the system in the simulation. Thus, it is way to acknowledge possible differences between the designed system and the simulated system, typically when it is too expensive to modify existing simulations. However, it also complicates configuration management.

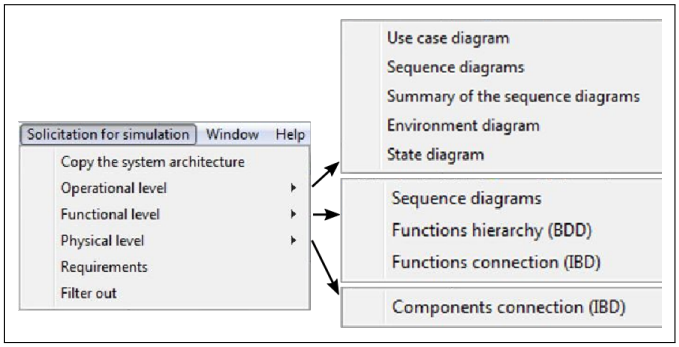

Fig. 6. Menus added to the top bar in PhiSystem.

After the copy, the system architect must define the perimeter and the level of detail he expects in the simulation. An important proposition is to select the elements to be simulated following steps similar to the design process. The operational level is first used to define the perimeter of the simulation, and the functional level is then used to define the level of detail of the simulation. The copied use case diagram can be opened thanks to the submenu “Operational level / Use case diagram”. As shown in Fig. 7, the use cases can be selected or unselected with a right-click menu (the figure has been edited to hide non-related menus). When a use case is selected, like “to pass a traffic light”, the selection is extended to all the use-cases it includes and all the use cases which include it. When a use case is unselected, the unselection is extended to all the use cases it includes, except if they are included in other selected use cases. The selected use cases appear in orange.

The tool directly looks for the related operational sequence diagrams. These sequence diagrams and a summary of these sequence diagrams can be opened with “Operational level” submenus. They allow the system architect to narrow the simulation perimeter by unselecting exchanges between the system and its environment. The summary is a table in a popup window where it is possible to unselect an individual message, all the messages of a diagram, or all the messages related to an element of the environment in all the diagrams. The next “Operational level” submenu shows the environment

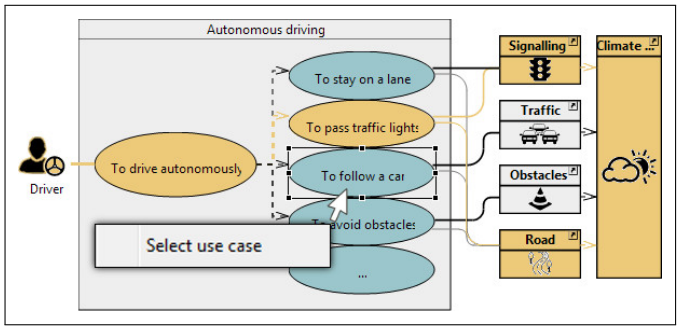

Fig. 7. Selection of use cases.

involved in the simulation, and the following one shows the states involved in the simulation (Fig. 8).

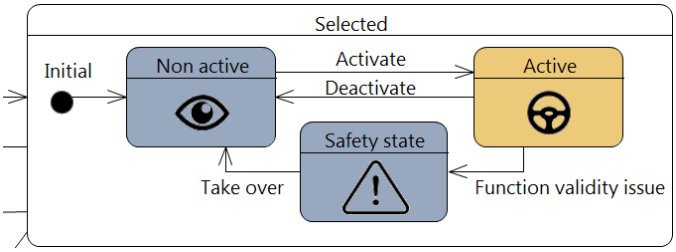

Fig. 8. Representation of the selected states.

When the user clicks on the submenu "Functional level / Sequence diagrams", the tool looks for and open the related functional sequence diagrams. The exchanges shown in these diagrams are selected for the simulation, except those which appear in the operational level and which have been unselected. The next "Functional level" submenu shows the functions hierarchy in a Block Definition Diagram (BDD). As shown in Fig. 9, a right click on a selected function permits to ignore its subfunctions and lower the level of detail of the simulation. For example, it is possible to ignore that the function "Navigate", corresponding to the sensors, has been designed with subfunctions such as "Acquire the shapes", "Acquire the motions" or "Merge the information". Finally, the last "Functional level" submenu shows the internal block diagram with the functions selected for the simulation.

The submenu "Requirements" opens a popup window which shows the requirements associated to all the functions and components properties. It first highlights the requirements directly coming from the needs of the stakeholders, like the electrical consumption of the car. It also highlights the requirements which arose during the design of the system and which still have to be associated to specific numerical values. The requirements from the needs represent the constraints of the problem, and the requirements with no specific numerical values represent the unknowns. It is possible to unselect the requirements from the needs of specific services like "acoustics" if they are out of the simulation's scope.

Finally, the submenu "Filter out" deletes all the elements which have not been selected for the simulation. This deletion

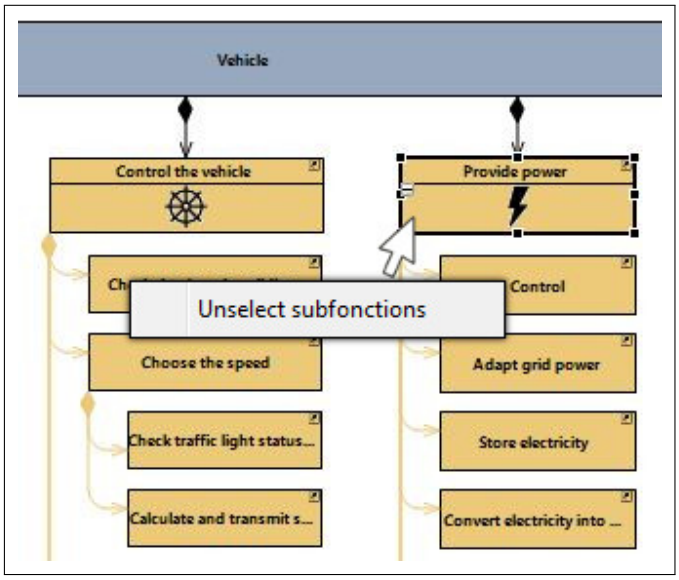

Fig. 9. Unselection of subfunctions in the BDD.

is done in the copy of the system architecture and keeps the original system architecture unchanged.

B. Resulting question

The filtered requirements help to formulate the question as they show the values which still have to be set, like the control and sensor properties, as well as the values constrained by stakeholder needs, like the cost or the electrical consumption. Thus, the system architect can easily ask for an optimization of the control and sensor properties to minimize the cost and electrical consumption. The related software functions will be part of a later article.

C. Environment scenarios to be tested

By definition, the environment, which is not under control, is associated to uncertainty. For example, the system architecture specifies how the system behaves if it faces a traffic light, but the traffic light is only a hypothetical event. Many uncertainties are implicit, like the location of the traffic lights. In a simulation, these uncertainties can either be defined in an explicit form, or be reduced to specific values in order to keep the number of simulation runs under an acceptable threshold. For example, the simulation can be done with a traffic light at exactly 500 meters.

The definition of the environmental scenarios is progressively refined, from the general point of view of the system architect which can choose a traffic light at 500 meters along a straight road, to the more accurate points of view of the simulation architect and experts. The related software functions will be part of a later article.

IV. Simulation architecture

The solicitation from the system architect is handled by a simulation architect. The simulation architect has a role similar to the system architect, but instead of specifying components

to answer customer needs, he specifies simulation models and software functions to answer the solicitation from the system architect. It is chosen to describe the simulation models with the properties of the "Model Identity Card" (MIC) which was initially defined in a PhD thesis at Renault in interaction with IRT SystemX ([23]). The MIC and FMI/FMU ([24]) have some properties in common, but the MIC aims at specifying or describing a simulation model rather than executing it. Thus, [25] presents a software function converting a MIC into an empty Modelica model, but this model then needs to be filled with equations in order to be executed.

It is also chosen to represent the simulation models with SysML blocks. These blocks are associated to the MIC properties through a stereotype defined in the profile diagram of Fig. 10.

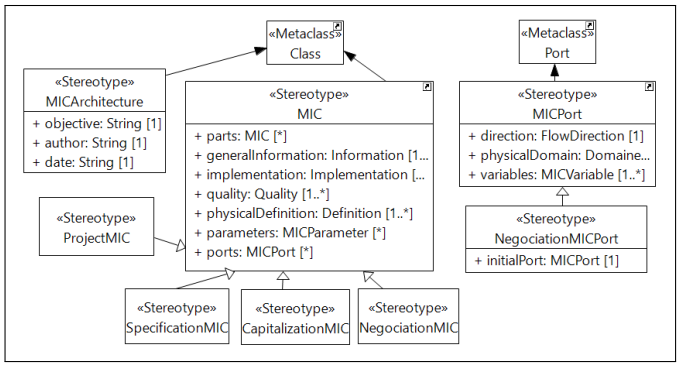

Fig. 10. The profile used for the MIC in PhiSystem.

Working in a SysML environment makes it convenient to map the simulation to the system it represents, with two major benefits. First, if a library archives the systems represented in the past simulations, it is possible to identify those similar to the solicitation, and re-use the corresponding simulation models. Second, it permits to validate the correct representation of the system in the simulation. As shown in the data model of Fig. 11, it is chosen to map: 1) the simulation model to the system function it represents 2) the ports and variables. A simulation model can be directly mapped to a system component fulfilling different functions.

When a simulation model exists, the information describing the simulation model is gathered in a "capitalization MIC". When an existing simulation model can be re-used after som e modifications to solve some inconsistencies with the solicitation, the simulation architect can change the "capitalization MIC" into a "negotiation MIC" to specify these modifications. Fig. 12 shows an example of negotiation MIC for an existing simulation model (on the right) based on the mapping with a system function (on the left) related to the vehicle's control. In the simulation model, the ports in white are consistent with the system function,the ports in grey are not, and the ports in red are specifications of new ports replacing the ports in grey. Thus, the port "Sensor", which was covering very different ports of the system function, is split into two.

The simulation architect can finally specify a new model, to be developed from scratch, in a "specification MIC".

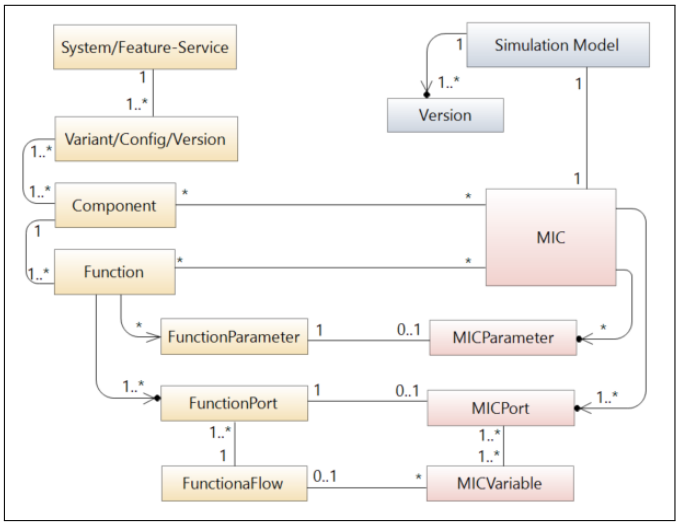

Fig. 11. The MIC data model.

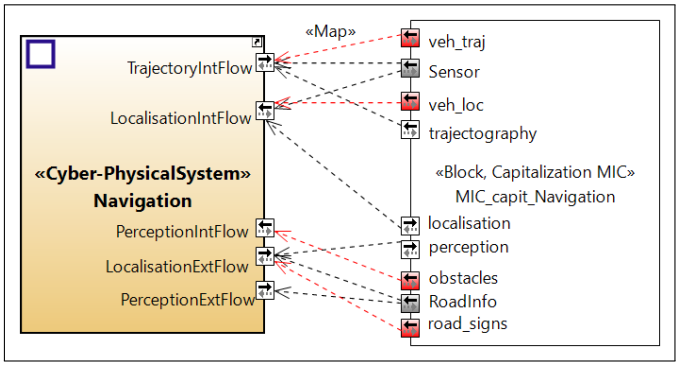

Fig. 12. Mapping of a function to a MIC in PhiSystem.

The interface of the specification MIC, and in particular its ports, can be generated from the function (or component) it represents in the solicitation.

All the simulation models are integrated in an internal block diagram, whether they are associated to a capitalization, a negotiation, or a specification MIC. The resulting architecture of MICs, illustrated in Fig. 13, represent the system and the environment defined in the solicitation. The MIC architecture has global inputs and outputs. In the problem of the autonomous vehicle passing traffic lights, the global inputs are the control and sensor properties to be optimized, as well as the environment properties to be changed in the different scenarios. The global outputs, which typically are functions of time, include the state of charge of the battery and the position of the vehicle which permit together to calculate the electrical consumption of the vehicle.

The architecture of MICs is completed by pre and post-treatment as well as non-physical black boxes, like an economic model outputting a cost, to finally form the simulation architecture represented in Fig. 14. Indeed, the architecture of MICs cannot reach alone the objective set by the system architect, i.e. optimizing the control and sensor properties to minimize the cost and electrical consumption. Each block of

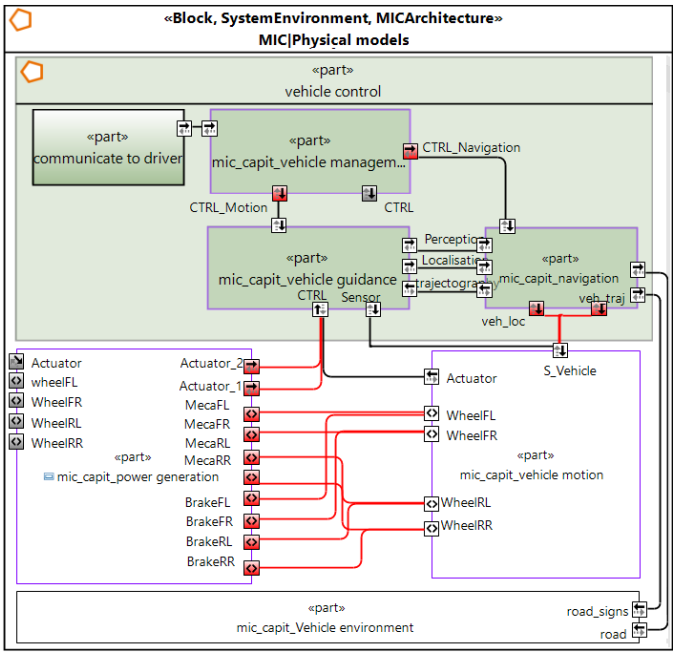

Fig. 13. Architecture of MICs for the system and environment.

the simulation architecture is allocated to software resources (such as Amesim or Simulink), and each software resource is allocated to hardware resources (such as a computer or a server), thus leading to a three-level representation of the simulation. It is for example possible to check that the resources respect the implementation constrains described in each MIC (e.g. regarding the version of Simulink to use), and to estimate the cost and performance of these resources. The different tasks described in this section will be supported by new software functions for the simulation architect.

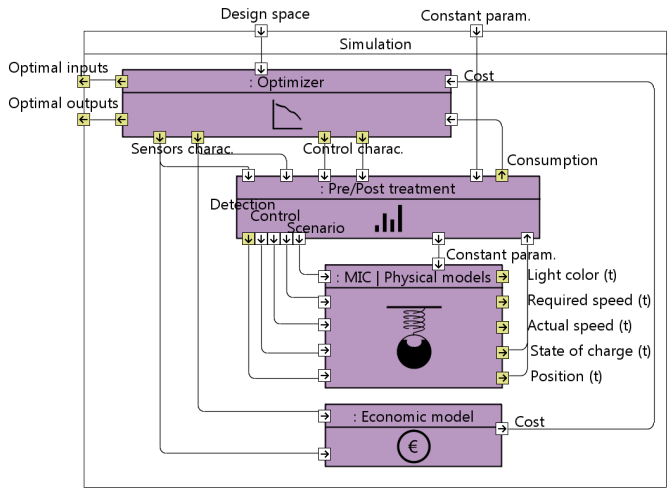

Fig. 14. Top-level view of the simulation architecture.

V. Numerical Simulation

After the design of the simulation architecture by the simulation architect, the missing models are developed by the experts of different simulation areas. The simulation models are then integrated and the simulation is executed. In the industrial design problem, the functions to sense and control are simulated using Matlab-Simulink, while the powertrain and the vehicle energetics are simulated using a library of Amesim. Matlab-Simulink and Amesim are integrated in a co-simulation illustrated in Fig. 15. This co-simulation is called at each iteration of the optimization process to evaluate candidate solutions. We consider a multiobjective optimizer as we aim at minimizing two conflictual objectives, namely: the cost and the electrical consumption averaged over multiple traffic lights. Among several state-of-the-art algorithms, we choose the NSGA-II (Non dominated Sorting Genetic Algorithm [26]) which is one of the most popular and widely used algorithms in the field of evolutionary multiobjective optimization. Experiments were conducted using the following settings. Four design parameters were considered. The first two are related to the control and were presented in section II : the distance from which the traffic light starts to be taken into account, and the maximum acceptable deceleration. The two others permit to describe the uncertain distance from which the traffic light is detected: the detection mode and the scale of a Gumbel probabilistic law. The population size was set to fifty. The optimization was stopped after fifty iterations.

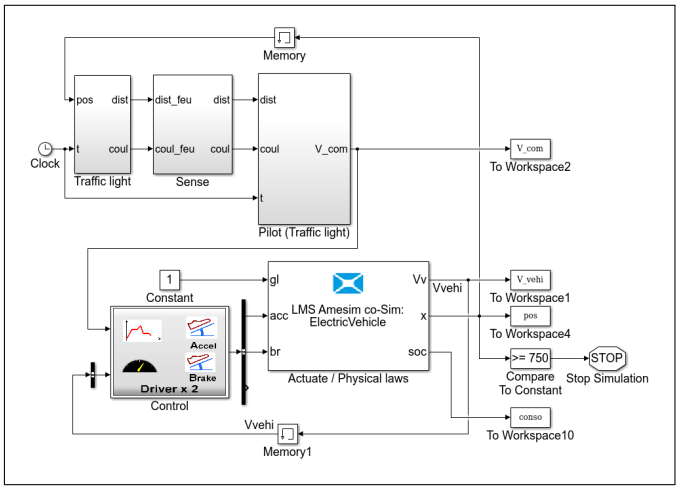

Fig. 15. Simulink view of the co-simulation (with Amesim).

After running the optimization, the simulation architect provides post-treated results to the system architect in order to take decisions. In our example, the returned results are represented by a Pareto front that illustrates the trade-off between optimal solutions (see Fig.16.)

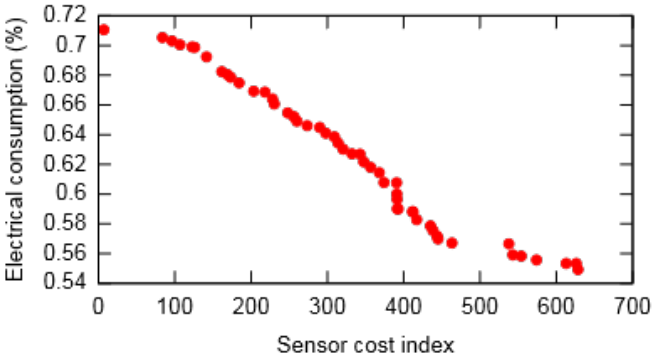

Fig. 16. Optimization results : Pareto Front.

VI. Conclusion

This work further bridges the gap between system architecture and simulation. Two tasks carried out by two different roles are clearly distinguished. First, the system architect formulates a solicitation for a simulation by using the part of the system architecture associated to specific use cases. A Java plugin has been implemented in a SysML editor based on Papyrus to allow the system architect to define the perimeter and the level of detail of the simulation, sparing him from the tedious processing of SysML data. The definition of a question based on the system requirements and the choice of an environment scenario have been explored and will also be facilitated by the plugin in the future. The second task considered in this work is the design of a simulation architecture by a simulation architect. A methodology supported by a metamodel has been developed to specify the connections between the simulation models, to identify their relations with the systems functions and components, and to solve any inconsistency coming from the re-use of former simulation models or from the lack of communication between the system architect and the simulation experts. The solutions presented in this work have been defined and validated thanks to the industrial design problem of an autonomous vehicle passing traffic lights. These solutions facilitate Agile project management by permitting frequent simulation loops and by improving the capitalization and re-use of simulation models. They will be combined with additional software developments whose objective is to improve the decision of the architects, when looking for former simulation models or when reviewing the simulation results for example.

References

[1] F. Retho, "Collaborative methodology for virtual product building to support aerial vehicles with electrical propulsion design," Theses, Supélec, May 2015.

[2] A. Forrai, *Embedded Control System Design: A Model Based Approach*. Springer Science & Business Media, 2012.

[3] F. Bordeleau and E. Fiallos, "Model-based engineering: A new era based on Papyrus and open source tooling." in *OSS4MDE@ MoDELS*. Citeseer, 2014, pp. 2–8.

[4] "No Magic acquisition completed," https://bit.ly/2T0GB2w accessed Feb. 10, 2019.

[5] "CIL4Sys Engineering," http://cil4sys.com/ accessed Feb. 10, 2019.

[6] "Syscience," https://www.syscience.fr/ accessed Feb. 10, 2019.

[7] MIT, "Architecture and systems engineering," https://sysengonline.mit.edu/ accessed Feb. 10, 2019.

[8] CESAMES, "Formation d'introduction à l'architecture," https://bit.ly/2qBiASO accessed Feb. 10, 2019.

[9] C. Yang, P. Ménégazzi, J.-D. Piques, O. Coppin, P. Chesse, and D. Chalet, "MBSE approach adapted to vehicle energy consumption optimization," in *NAFEMS World Congress 2017*, 2017.

[10] P. Roques, "MBSE with the ARCADIA method and the Capella tool," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[11] PSA Groupe, "Challenges et enjeux de la simulation numérique," 2017, https://bit.ly/2QxcsWY accessed Feb. 10, 2019.

[12] Renault, "Chef d'équipe Model Factory," https://bit.ly/2FmSNbs accessed Feb. 10, 2019.

[13] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[14] Argosim, "STIMULUS for requirements," https://bit.ly/2JXl32F accessed Feb. 10, 2019.

[15] A. Rauzy, "Five theses for model-based systems engineering and model-based safety assessment," Norwegian University of Science and Technology, Tech. Rep., 2016.

[16] E. Huang, R. Ramamurthy, and L. F. McGinnis, "System and simulation modeling using SysML," in *2007 Winter Simulation Conference*, Dec 2007, pp. 796–803.

[17] Y. Cao, Y. Liu, H. Fan, and B. Fan, "SysML-based uniform behavior modeling and automated mapping of design and simulation model for complex mechatronics," *Computer-Aided Design*, vol. 45, no. 3, pp. 764–776, 2013.

[18] R. S. Peak, R. M. Burkhart, S. A. Friedenthal, M. W. Wilson, M. Bajaj, and I. Kim, "Simulation-based design using SysML - Part 1: A parametrics primer," in *INCOSE international symposium*, vol. 17, no. 1. Wiley Online Library, 2007, pp. 1516–1535.

[19] ——, "Simulation-based design using SysML - Part 2: Celebrating diversity by example," in *INCOSE International Symposium*, vol. 17, no. 1. Wiley Online Library, 2007, pp. 1536–1557.

[20] Phoenix Integration, "ModelCenter MBSEPak," https://bit.ly/2AngbzA accessed Feb. 10, 2019.

[21] C. J. Paredis and T. Johnson, "Using OMG's SysML to support simulation," in *Proceedings of the 40th Conference on Winter Simulation*. Winter Simulation Conference, 2008, pp. 2350–2352.

[22] R. Renier and R. Chenouard, "De SysML à Modelica : Aide à la formalisation de modèles de simulation en conception préliminaire," in *12ème Colloque National AIP PRIMECA*, 2011.

[23] G. Sirin, "Supporting multidisciplinary vehicle modeling: towards an ontology-based knowledge sharing in collaborative model based systems engineering environment," Ph.D. dissertation, Châtenay-Malabry, Ecole centrale de Paris, 2015.

[24] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, H. Elmqvist, A. Junghanns, J. Mauß, M. Monteiro, T. Neidhold, D. Neumerkel *et al.*, "The functional mockup interface for tool independent exchange of simulation models," in *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*, no. 063. Linköping University Electronic Press, 2011, pp. 105–114.

[25] G. Fontaine, "Modélisation théorique et processus associés pour architectes modèle dans un environnement multidisciplinaire," Ph.D. dissertation, Paris Saclay, 2017.

[26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.